In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.feature_selection import VarianceThreshold
variance=VarianceThreshold(threshold=0)
from sklearn.preprocessing import LabelEncoder
label=LabelEncoder
```

In [2]:
```
train=pd.read_csv('train.csv')
train.head()
```

Out[2]:

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	...	X375	X376	X377	X378	X379	X380	X382	X385
0	0	130.81	k	v	at	a	d	u	j	o	...	0	0	1	0	0	0	0	0
1	6	88.53	k	t	av	e	d	y	l	o	...	1	0	0	0	0	0	0	0
2	7	76.26	az	w	n	c	d	x	j	x	...	0	0	0	0	0	0	1	0
3	9	80.62	az	t	n	f	d	x	l	e	...	0	0	0	0	0	0	0	0
4	13	78.02	az	v	n	f	d	h	d	n	...	0	0	0	0	0	0	0	0

5 rows × 378 columns

In [3]:
```
test=pd.read_csv('test.csv')
test.head()
```

Out[3]:

	ID	X0	X1	X2	X3	X4	X5	X6	X8	X10	...	X375	X376	X377	X378	X379	X380	X382	X
0	1	az	v	n	f	d	t	a	w	0	...	0	0	0	1	0	0	0	
1	2	t	b	ai	a	d	b	g	y	0	...	0	0	1	0	0	0	0	
2	3	az	v	as	f	d	a	j	j	0	...	0	0	0	1	0	0	0	
3	4	az	l	n	f	d	z	l	n	0	...	0	0	0	1	0	0	0	
4	5	w	s	as	c	d	y	i	m	0	...	1	0	0	0	0	0	0	

5 rows × 377 columns

In [4]:
```
test.describe()
```

Out[4]:

	ID	X10	X11	X12	X13	X14	X15	
count	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4
mean	4211.039202	0.019007	0.000238	0.074364	0.061060	0.427893	0.000713	
std	2423.078926	0.136565	0.015414	0.262394	0.239468	0.494832	0.026691	
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	2115.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
50%	4202.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
75%	6310.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	
max	8416.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	

8 rows × 369 columns

In [5]:
```
test.describe()
```

Out[5]:

	ID	X10	X11	X12	X13	X14	X15	
count	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4
mean	4211.039202	0.019007	0.000238	0.074364	0.061060	0.427893	0.000713	
std	2423.078926	0.136565	0.015414	0.262394	0.239468	0.494832	0.026691	
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	2115.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
50%	4202.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
75%	6310.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	
max	8416.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	

8 rows × 369 columns

In [6]:
```
train.isnull().sum()
```

Out[6]:
```
ID      0
y       0
X0      0
X1      0
X2      0
       ..
X380    0
X382    0
X383    0
X384    0
X385    0
Length: 378, dtype: int64
```

In [7]:
```
train_target=train['y']
train_data=train.drop(['y','ID'],axis=1)
```

```
train_data.head(4)
```

Out[7]:

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X11	...	X375	X376	X377	X378	X379	X380	X382
0	k	v	at	a	d	u	j	o	0	0	...	0	0	1	0	0	0	0
1	k	t	av	e	d	y	l	o	0	0	...	1	0	0	0	0	0	0
2	az	w	n	c	d	x	j	x	0	0	...	0	0	0	0	0	0	1
3	az	t	n	f	d	x	l	e	0	0	...	0	0	0	0	0	0	0

4 rows × 376 columns

In [8]:
```
train_data.var().sort_values().head(10)
```

```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_5736\1341674288.py:1: FutureWarning: Dro
pping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is depre
cated; in a future version this will raise TypeError.  Select only valid columns befo
re calling the reduction.
  train_data.var().sort_values().head(10)
```

Out[8]:
```
X330    0.0
X297    0.0
X268    0.0
X290    0.0
X235    0.0
X347    0.0
X107    0.0
X233    0.0
X289    0.0
X93     0.0
dtype: float64
```

In [9]:
```
train_data_not_null=variance.fit_transform(train_data.iloc[:,9:])
```

In [10]:
```
train_data_not_null
```

Out[10]:
```
array([[0, 1, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [1, 1, 0, ..., 0, 0, 0],
       [0, 0, 1, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

In [11]:
```
labeled_data=train_data.iloc[:,0:8]
```

In [12]:
```
labeled_data.head()
```

Out[12]:

In [13]:
```
labeled_data.nunique()
```

Out[13]:
```
X0    47
X1    27
X2    44
X3     7
X4     4
X5    29
X6    12
X8    25
dtype: int64
```

In [14]:
```
labeled_data1=labeled_data.apply(label().fit_transform)
labeled_data1.head()
```

Out[14]:

	X0	X1	X2	X3	X4	X5	X6	X8
0	32	23	17	0	3	24	9	14
1	32	21	19	4	3	28	11	14
2	20	24	34	2	3	27	9	23
3	20	21	34	5	3	27	11	4
4	20	23	34	5	3	12	3	13

In [15]:
```
labeled_data1.var()
```

Out[15]:
```
X0    188.741938
X1     72.777974
X2    118.808135
X3      3.027295
X4      0.005461
X5     68.076236
X6      8.508730
X8     49.531868
dtype: float64
```

In [16]:
```
train_data_not_null_final=pd.DataFrame(train_data_not_null)
train_data_not_null_final
```

Out[16]:

	0	1	2	3	4	5	6	7	8	9	...	345	346	347	348	349	350	351	352	353	354
0	0	1	0	0	0	0	1	0	0	1	...	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0	...	1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	1	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4204	0	0	1	0	0	0	0	0	0	0	...	1	0	0	0	0	0	0	0	0	0
4205	0	0	0	0	0	0	0	0	0	0	...	0	1	0	0	0	0	0	0	0	0
4206	1	1	0	0	0	0	0	0	0	0	...	0	0	1	0	0	0	0	0	0	0
4207	0	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4208	0	0	0	0	0	0	0	0	1	0	...	1	0	0	0	0	0	0	0	0	0

4209 rows × 355 columns

In [17]:
```
final_train=pd.concat([labeled_data1,train_data_not_null_final],axis=1)
final_train.head()
```

Out[17]:

	X0	X1	X2	X3	X4	X5	X6	X8	0	1	...	345	346	347	348	349	350	351	352	353	354
0	32	23	17	0	3	24	9	14	0	1	...	0	0	1	0	0	0	0	0	0	0
1	32	21	19	4	3	28	11	14	0	0	...	1	0	0	0	0	0	0	0	0	0
2	20	24	34	2	3	27	9	23	0	0	...	0	0	0	0	0	0	1	0	0	0
3	20	21	34	5	3	27	11	4	0	0	...	0	0	0	0	0	0	0	0	0	0
4	20	23	34	5	3	12	3	13	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 363 columns

In [18]:
```
final_train.isnull().any()
```

Out[18]:
```
X0      False
X1      False
X2      False
X3      False
X4      False
        ...
350     False
351     False
352     False
353     False
354     False
Length: 363, dtype: bool
```

In [19]:
```
test=test.drop(['ID'],axis=1)
test.head()
```

Out[19]:

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X11	...	X375	X376	X377	X378	X379	X380	X382
0	az	v	n	f	d	t	a	w	0	0	...	0	0	0	1	0	0	0
1	t	b	ai	a	d	b	g	y	0	0	...	0	0	1	0	0	0	0
2	az	v	as	f	d	a	j	j	0	0	...	0	0	0	1	0	0	0
3	az	l	n	f	d	z	l	n	0	0	...	0	0	0	1	0	0	0
4	w	s	as	c	d	y	i	m	0	0	...	1	0	0	0	0	0	0

5 rows × 376 columns

In [20]:
```
test.nunique()
```

Out[20]:
```
X0      49
X1      27
X2      45
X3       7
X4       4
        ..
X380     2
X382     2
X383     2
X384     2
X385     2
Length: 376, dtype: int64
```

In [21]:
```
test.isnull().any()
```

Out[21]:
```
X0      False
X1      False
X2      False
X3      False
X4      False
        ...
X380    False
X382    False
X383    False
X384    False
X385    False
Length: 376, dtype: bool
```

In [22]:
```
test_not_null=variance.transform(test.iloc[:,9:])
test_not_null
```

Out[22]:
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 1, ..., 0, 0, 0],
       ...,
       [0, 0, 1, ..., 0, 0, 0],
       [0, 1, 1, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

In [23]:
```
test_not_null_final=pd.DataFrame(test_not_null)
test_not_null_final.head()
```

Out[23]:

	0	1	2	3	4	5	6	7	8	9	...	345	346	347	348	349	350	351	352	353	354
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	0	0	...	0	0	1	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0	...	0	0	0	1	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	1	0	0	0	0	0	0
4	0	0	1	0	0	0	0	0	0	0	...	1	0	0	0	0	0	0	0	0	0

5 rows × 355 columns

In [24]:
```
labeled_data1=test.iloc[:,0:8]
labeled_data1.head()
```

Out[24]:

	X0	X1	X2	X3	X4	X5	X6	X8
0	az	v	n	f	d	t	a	w
1	t	b	ai	a	d	b	g	y
2	az	v	as	f	d	a	j	j
3	az	l	n	f	d	z	l	n
4	w	s	as	c	d	y	i	m

In [25]:
```
test_label=labeled_data.apply(label().fit_transform)
test_label.head()
```

Out[25]:

	X0	X1	X2	X3	X4	X5	X6	X8
0	32	23	17	0	3	24	9	14
1	32	21	19	4	3	28	11	14
2	20	24	34	2	3	27	9	23
3	20	21	34	5	3	27	11	4
4	20	23	34	5	3	12	3	13

In [26]:
```
test_data_final=pd.concat([test_label,test_not_null_final],axis=1)
test_data_final.head()
```

Out[26]:

	X0	X1	X2	X3	X4	X5	X6	X8	0	1	...	345	346	347	348	349	350	351	352	353	354
0	32	23	17	0	3	24	9	14	0	0	...	0	0	0	1	0	0	0	0	0	0
1	32	21	19	4	3	28	11	14	0	0	...	0	0	1	0	0	0	0	0	0	0
2	20	24	34	2	3	27	9	23	0	0	...	0	0	0	1	0	0	0	0	0	0
3	20	21	34	5	3	27	11	4	0	0	...	0	0	0	1	0	0	0	0	0	0
4	20	23	34	5	3	12	3	13	0	0	...	1	0	0	0	0	0	0	0	0	0

5 rows × 363 columns

In [27]:
```
##performing Dimensionality reduction using principal component analysis
from sklearn.model_selection import train_test_split
```

In [28]:
```
x_train,x_test,y_train,y_test=train_test_split(final_train,train_target,random_state=4
```

In [29]:
```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Out[29]:
```
((2946, 363), (1263, 363), (2946,), (1263,))
```

In [30]:
```
from sklearn.decomposition import PCA
pca=PCA(n_components=2)
```

In [31]:
```
x_train=pca.fit_transform(x_train)
x_test=pca.transform(x_test)
test_data_final=pca.transform(test_data_final)
```

```
C:\Users\Lenovo\anaconda3\lib\site-packages\sklearn\utils\validation.py:1688: FutureW
arning: Feature names only support names that are all strings. Got feature names with
dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
C:\Users\Lenovo\anaconda3\lib\site-packages\sklearn\utils\validation.py:1688: FutureW
arning: Feature names only support names that are all strings. Got feature names with
dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
C:\Users\Lenovo\anaconda3\lib\site-packages\sklearn\utils\validation.py:1688: FutureW
arning: Feature names only support names that are all strings. Got feature names with
dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

In [33]:
```
## XGBaost
from sklearn import svm
from sklearn.metrics import r2_score,mean_squared_error
from xgboost import XGBRegressor
xgbr=XGBRegressor(randome_state=42)
```

In [38]:
```
model=xgbr.fit(x_train,y_train)
```

```
[17:53:20] WARNING: C:/Users/administrator/workspace/xgboost-win64_release_1.6.0/src/
learner.cc:627:
Parameters: { "randome_state" } might not be used.

  This could be a false alarm, with some parameters getting used by language bindings
but
  then being mistakenly passed down to XGBoost core, or some parameter actually being
used
  but getting flagged wrongly here. Please open an issue if you find any such cases.
```

In [41]:
```
ypred_test=model.predict(x_test)
ypred_test
```

Out[41]:
```
array([ 89.92478,  92.23022, 106.76723, ...,  93.1157 , 100.5901 ,
       107.29518], dtype=float32)
```

In [42]:
```
ypred_train=model.predict(x_train)
```

```
ypred_train
```

Out[42]:
```
array([ 93.82634 , 111.754364, 109.68195 , ...,  99.46664 ,  93.587746,
        94.289566], dtype=float32)
```

In [43]:
```
print(r2_score(ypred_train,y_train))
```

```
0.6854826904515524
```

In [44]:
```
print(mean_squared_error(ypred_train,y_train))
```

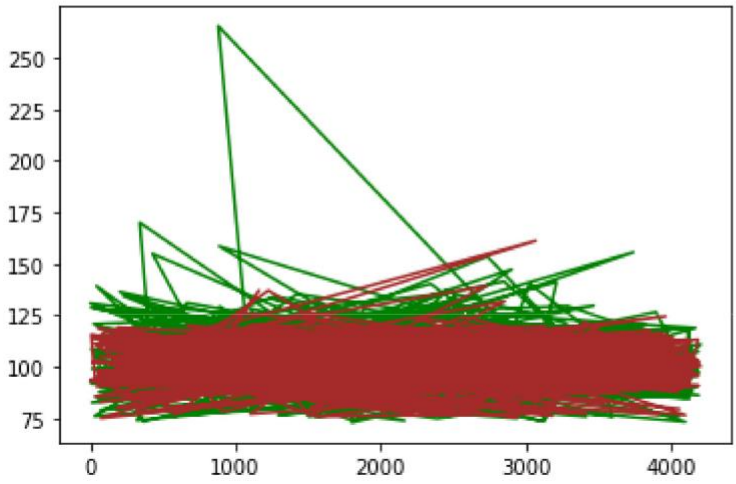
```
28.587957391170832
```

In [45]:
```
test_data_final_prediction=model.predict(test_data_final)
test_data_final_prediction
```

Out[45]:
```
array([ 92.837296, 104.064445,  80.51412 , ..., 107.84169 ,  94.63862 ,
       100.92952 ], dtype=float32)
```

In [46]:
```
prediction=pd.DataFrame({'ytest':y_test,'ypred':ypred_test})
```

In [47]:
```
plt.plot(prediction['ytest'],color='green')
plt.plot(prediction['ypred'],color='brown')
plt.show()
```



In [ ]:
```

```